Actividad 7 (Evaluación Final)

Bruno Manuel Zamora Garcia A01798275

En este robot de 6 GDL:

- Las juntas 1, 2, 4, 5, 6 son rotacionales.
- La junta 3 es prismática.El objetivo principal es:

1. Calcular la energía cinética total KTotal.
2. Calcular la energía potencial total UTotal.
3. Obtener el Lagrangiano L=K−U.
4. Explicar por qué, a pesar de que la junta 3 es prismática, el eslabón 3 presenta una velocidad angular W3≠0 (vector diferente de [0 0 0]).

El código en MATLAB realiza estos cálculos de forma simbólica: define las variables, construye la cinemática directa, obtiene los Jacobianos lineal y angular de cada eslabón y finalmente calcula las energías cinética y potencial.

```
clear all; close all; clc;
tic;

%-------------------------------------------------------------------------
% Declaración de variables simbólicas
%-------------------------------------------------------------------------
% Defino las variables de desplazamiento/ángulo y sus derivadas
syms th1(t) th2(t) th3(t) th4(t) th5(t) th6(t) t
syms th1p(t) th2p(t) th3p(t) th4p(t) th5p(t) th6p(t)
syms th1pp(t) th2pp(t) th3pp(t) th4pp(t) th5pp(t) th6pp(t)

% Defino las masas e inercia de cada eslabón
syms m1 m2 m3 m4 m5 m6
syms Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2 Ixx3 Iyy3 Izz3 Ixx4 Iyy4 Izz4 Ixx5 Iyy5
Izz5 Ixx6 Iyy6 Izz6

% Variables de longitud o recorrido y sus distancias al centro de masa
syms d1 d2 l3 d4 d5 d6 dc1 dc2 lc3 dc4 dc5 dc6

% Constantes físicas
syms pi g a cero

%-------------------------------------------------------------------------
% Vectores articulares
%-------------------------------------------------------------------------
% Creo los vectores de coordenadas (Q), velocidades (Qp) y aceleraciones
(Qpp)
Q   = [th1; th2; th3; th4; th5; th6];
Qp  = [th1p; th2p; th3p; th4p; th5p; th6p];
```

```
Qpp = [th1pp; th2pp; th3pp; th4pp; th5pp; th6pp];
```

2.Configuración del Robot y Cinemática Directa

1.Se define el vector `RP = [0 0 1 0 0 0]`, donde `0` significa junta rotacional y `1` junta prismática (la 3).

2.Se construye la cinemática directa mediante la definición de:

- P(:,:,i): la posición de la articulación iii.
- R(:,:,i): la matriz de rotación de la articulación iii.

3.A partir de estas, se generan las matrices de transformación homogénea Ai y se acumulan en las transformaciones globales Ti.

```
%-------------------------------------------------------------------------
% Configuración del robot
%-------------------------------------------------------------------------
% 0 -> rotacional, 1 -> prismática
% La tercera articulación es prismática
RP = [0 0 1 0 0 0];
GDL = size(RP,2);  % Total de grados de libertad

%-------------------------------------------------------------------------
% Definición de la Cinemática Directa
%-------------------------------------------------------------------------
% Asigno las posiciones (P) y rotaciones (R) de cada articulación
P(:,:,1) = [0; 0; d1];
R(:,:,1) = rotacion_z(90)*rotacion_x(-90);

P(:,:,2) = [0; 0; d2];
R(:,:,2) = rotacion_z(90)*rotacion_x(90);

P(:,:,3) = [0; 0; l3];
R(:,:,3) = eye(3);

P(:,:,4) = [0; 0; d4];
R(:,:,4) = rotacion_z(90)*rotacion_x(-90);

P(:,:,5) = [0; 0; d5];
R(:,:,5) = rotacion_z(90) * rotacion_x(90);

P(:,:,6) = [0; 0; d6];
R(:,:,6) = eye(3);

%-------------------------------------------------------------------------
% Construcción de las Matrices de Transformación Homogénea
%-------------------------------------------------------------------------
% Vector para completar la última fila [0 0 0 1]
Vector_Zeros = zeros(1, 3);

% Inicializo la matriz local y global para el último eslabón
```

```
A(:,:,GDL)  = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros, 1]);
T(:,:,GDL)  = simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros, 1]);
PO(:,:,GDL) = P(:,:,GDL);
RO(:,:,GDL) = R(:,:,GDL);

% Calculo las transformaciones globales multiplicando secuencialmente
for i = 1:GDL
    A(:,:,i) = simplify([R(:,:,i) P(:,:,i); Vector_Zeros, 1]);
    try
       T(:,:,i) = T(:,:,i-1) * A(:,:,i);
    catch
       T(:,:,i) = A(:,:,i);
    end
    T(:,:,i) = simplify(T(:,:,i));
    RO(:,:,i) = T(1:3,1:3,i);
    PO(:,:,i) = T(1:3,4,i);
end
```

3. Cálculo de Jacobianos y Velocidades

1. Para cada eslabón i, se calcula el Jacobiano lineal Jv y el Jacobiano angular Jwa, teniendo en cuenta todas las juntas anteriores.
2. Las juntas prismáticas aportan Jw=0(no hay giro propio) y Jv=zk−1
3. Las juntas rotacionales aportan Jv=zk−1×(pfinal−pk−1) y Jw=zk−1.
4. Se multiplica cada Jacobiano por el vector de velocidades Qp para obtener: Vi=JvaQp, Wi=JwaQp.

Nota sobre la velocidad angular del Eslabón 3

- Aunque la junta 3 sea prismática, el eslabón 3 hereda las rotaciones de las juntas 1 y 2 (que son rotacionales). Por ello, W3 parece con términos en θ1′ y θ2′.
- La junta 3 no añade giro propio, pero el eslabón 3 sí rota debido a las articulaciones anteriores.

```
%-----------------------------------------------------------------------
% Cálculo de los Jacobianos y Velocidades para cada Eslabón
%-----------------------------------------------------------------------
% Para cada eslabón, tomo en cuenta las juntas anteriores
% Si la junta es prismática, su contribución angular es cero

% Eslabón 6 (usa las 6 juntas)
Jv_a6(:,6) = PO(:,:,6);
Jw_a6(:,6) = PO(:,:,6);
for k = 1:6
    if RP(k) == 0
        try
            Jv_a6(:,k) = cross(RO(:,3,k-1), PO(:,:,6) - PO(:,:,k-1));
            Jw_a6(:,k) = RO(:,3,k-1);
        catch
            Jv_a6(:,k) = cross([0;0;1], PO(:,:,6));
            Jw_a6(:,k) = [0;0;1];
```

```
        end
    else
        try
            Jv_a6(:,k) = RO(:,3,k-1);
        catch
            Jv_a6(:,k) = [0;0;1];
        end
        Jw_a6(:,k) = [0;0;0];
    end
end
Jv_a6 = simplify(Jv_a6);
Jw_a6 = simplify(Jw_a6);
Jac6 = [Jv_a6; Jw_a6];
Jacobiano6 = simplify(Jac6);
Qp = Qp(t);  % Evalúo las velocidades como función de t

disp('Velocidad lineal del Eslabón 6 (efector final):');
```

```
Velocidad lineal del Eslabón 6 (efector final):
```

```
V6 = simplify(Jv_a6 * Qp);
pretty(V6);
```

```
/             d5 th4p(t) - th1p(t) (d4 + l3)            \
|                                                        |
| th3p(t) - th1p(t) (d2 + d6) + d5 th2p(t) - d6 th5p(t) |
|                                                        |
\             d6 th4p(t) - th2p(t) (d4 + l3)            /
```

```

disp('Velocidad angular del Eslabón 6 (efector final):');
```

```
Velocidad angular del Eslabón 6 (efector final):
```

```
W6 = simplify(Jw_a6 * Qp);
pretty(W6);
```

```
/ - th2p(t) - th6p(t) \
|                      |
|       th4p(t)        |
|                      |
\  th1p(t) + th5p(t)  /
```

```

% Eslabón 5 (usa 5 primeras juntas)
Jv_a5(:,5) = PO(:,:,5);
Jw_a5(:,5) = PO(:,:,5);
for k = 1:5
    if RP(k) == 0
        try
            Jv_a5(:,k) = cross(RO(:,3,k-1), PO(:,:,5) - PO(:,:,k-1));
            Jw_a5(:,k) = RO(:,3,k-1);
        catch
            Jv_a5(:,k) = cross([0;0;1], PO(:,:,5));
```

```
            Jw_a5(:,k) = [0;0;1];
        end
    else
        try
            Jv_a5(:,k) = RO(:,3,k-1);
        catch
            Jv_a5(:,k) = [0;0;1];
        end
        Jw_a5(:,k) = [0;0;0];
    end
end
Jv_a5 = simplify(Jv_a5);
Jw_a5 = simplify(Jw_a5);
Jac5 = [Jv_a5; Jw_a5];
Jacobiano5 = simplify(Jac5);

disp('Velocidad lineal del Eslabón 5:');
```

```
Velocidad lineal del Eslabón 5:
```

```
V5 = simplify(Jv_a5 * Qp(1:5));
pretty(V5);
```

```
/   d5 th4p(t) - th1p(t) (d4 + l3)  \
|                                    |
| th3p(t) - d2 th1p(t) + d5 th2p(t) |
|                                    |
\         -th2p(t) (d4 + l3)        /
```

```

disp('Velocidad angular del Eslabón 5:');
```

```
Velocidad angular del Eslabón 5:
```

```
W5 = simplify(Jw_a5 * Qp(1:5));
pretty(W5);
```

```
/      -th2p(t)     \
|                   |
|      th4p(t)      |
|                   |
\ th1p(t) + th5p(t) /
```

```

% Eslabón 4 (usa 4 primeras juntas)
Jv_a4(:,4) = PO(:,:,4);
Jw_a4(:,4) = PO(:,:,4);
for k = 1:4
    if RP(k) == 0
        try
            Jv_a4(:,k) = cross(RO(:,3,k-1), PO(:,:,4) - PO(:,:,k-1));
            Jw_a4(:,k) = RO(:,3,k-1);
        catch
            Jv_a4(:,k) = cross([0;0;1], PO(:,:,4));
```

```
            Jw_a4(:,k) = [0;0;1];
        end
    else
        try
            Jv_a4(:,k) = RO(:,3,k-1);
        catch
            Jv_a4(:,k) = [0;0;1];
        end
        Jw_a4(:,k) = [0;0;0];
    end
end
Jv_a4 = simplify(Jv_a4);
Jw_a4 = simplify(Jw_a4);
Jac4 = [Jv_a4; Jw_a4];
Jacobiano4 = simplify(Jac4);

disp('Velocidad lineal del Eslabón 4:');
```

```
Velocidad lineal del Eslabón 4:
```

```
V4 = simplify(Jv_a4 * Qp(1:4));
pretty(V4);
```

```
/  -th1p(t) (d4 + l3)  \
|                      |
| th3p(t) - d2 th1p(t) |
|                      |
\  -th2p(t) (d4 + l3)  /
```

```

disp('Velocidad angular del Eslabón 4:');
```

```
Velocidad angular del Eslabón 4:
```

```
W4 = simplify(Jw_a4 * Qp(1:4));
pretty(W4);
```

```
/ -th2p(t) \
|          |
|  th4p(t) |
|          |
\  th1p(t) /
```

```

% Eslabón 3 (usa 3 primeras juntas)
Jv_a3(:,3) = PO(:,:,3);
Jw_a3(:,3) = PO(:,:,3);
for k = 1:3
    if RP(k) == 0
        try
            Jv_a3(:,k) = cross(RO(:,3,k-1), PO(:,:,3) - PO(:,:,k-1));
            Jw_a3(:,k) = RO(:,3,k-1);
        catch
            Jv_a3(:,k) = cross([0;0;1], PO(:,:,3));
```

```
            Jw_a3(:,k) = [0;0;1];
        end
    else
        try
            Jv_a3(:,k) = RO(:,3,k-1);
        catch
            Jv_a3(:,k) = [0;0;1];
        end
        Jw_a3(:,k) = [0;0;0];
    end
end
Jv_a3 = simplify(Jv_a3);
Jw_a3 = simplify(Jw_a3);
Jac3 = [Jv_a3; Jw_a3];
Jacobiano3 = simplify(Jac3);

disp('Velocidad lineal del Eslabón 3:');
```

```
Velocidad lineal del Eslabón 3:
```

```
V3 = simplify(Jv_a3 * Qp(1:3));
pretty(V3);
```

```
/      -l3 th1p(t)     \
|                      |
| th3p(t) - d2 th1p(t) |
|                      |
\      -l3 th2p(t)     /
```

```

disp('Velocidad angular del Eslabón 3:');
```

```
Velocidad angular del Eslabón 3:
```

```
W3 = simplify(Jw_a3 * Qp(1:3));
pretty(W3);
```

```
/ -th2p(t) \
|          |
|     0    |
|          |
\  th1p(t) /
```

```

% Eslabón 2 (usa 2 primeras juntas)
Jv_a2(:,2) = PO(:,:,2);
Jw_a2(:,2) = PO(:,:,2);
for k = 1:2
    if RP(k) == 0
        try
            Jv_a2(:,k) = cross(RO(:,3,k-1), PO(:,:,2) - PO(:,:,k-1));
            Jw_a2(:,k) = RO(:,3,k-1);
        catch
            Jv_a2(:,k) = cross([0;0;1], PO(:,:,2));
```

```
            Jw_a2(:,k) = [0;0;1];
        end
    else
        try
            Jv_a2(:,k) = RO(:,3,k-1);
        catch
            Jv_a2(:,k) = [0;0;1];
        end
        Jw_a2(:,k) = [0;0;0];
    end
end
Jv_a2 = simplify(Jv_a2);
Jw_a2 = simplify(Jw_a2);
Jac2 = [Jv_a2; Jw_a2];
Jacobiano2 = simplify(Jac2);

disp('Velocidad lineal del Eslabón 2:');
```

```
Velocidad lineal del Eslabón 2:
```

```
V2 = simplify(Jv_a2 * Qp(1:2));
pretty(V2);
```

```
/      0      \
|             |
| -d2 th1p(t) |
|             |
\      0      /
```

```

disp('Velocidad angular del Eslabón 2:');
```

```
Velocidad angular del Eslabón 2:
```

```
W2 = simplify(Jw_a2 * Qp(1:2));
pretty(W2);
```

```
/ -th2p(t) \
|          |
|     0    |
|          |
\  th1p(t) /
```

```

% Eslabón 1 (usa solo la 1ra junta)
Jv_a1(:,1) = PO(:,:,1);
Jw_a1(:,1) = PO(:,:,1);
for k = 1:1
    if RP(k) == 0
        try
            Jv_a1(:,k) = cross(RO(:,3,k-1), PO(:,:,1) - PO(:,:,k-1));
            Jw_a1(:,k) = RO(:,3,k-1);
        catch
            Jv_a1(:,k) = cross([0;0;1], PO(:,:,1));
```

```
            Jw_a1(:,k) = [0;0;1];
        end
    else
        try
            Jv_a1(:,k) = RO(:,3,k-1);
        catch
            Jv_a1(:,k) = [0;0;1];
        end
        Jw_a1(:,k) = [0;0;0];
    end
end
Jv_a1 = simplify(Jv_a1);
Jw_a1 = simplify(Jw_a1);
Jac1 = [Jv_a1; Jw_a1];
Jacobiano1 = simplify(Jac1);

disp('Velocidad lineal del Eslabón 1:');
```

```
Velocidad lineal del Eslabón 1:
```

```
V1 = simplify(Jv_a1 * Qp(1:1));
pretty(V1);
```

```
/ 0 \
|   |
| 0 |
|   |
\ 0 /
```

```


disp('Velocidad angular del Eslabón 1:');
```

```
Velocidad angular del Eslabón 1:
```

```
W1 = simplify(Jw_a1 * Qp(1:1));
pretty(W1);
```

```
/    0    \
|         |
|    0    |
|         |
\ th1p(t) /
```

4.Cálculo de la Energía Cinética

1. Se sustituye la longitud di por lipor la distancia al centro de masa dcio lci en la posición P(:,:,i)
2. La energía cinética de cada eslabón iii se compone de: Ki = 0.5mi ∥VTotal,i∥2 + 0.5Wi*Ii*Wi donde VTotal,i=Vi+Wi×rcm*i
3. Se suman los Ki para obtener la energía cinética total KTotal=∑Ki

```
%-----------------------------------------------------------------------
% Cálculo de la Energía Cinética
%-----------------------------------------------------------------------
% Sustituyo las variables de recorrido por las distancias al centro de masa
```

```
P01 = subs(P(:,:,1), d1, dc1);
P02 = subs(P(:,:,2), d2, dc2);
P03 = subs(P(:,:,3), l3, lc3);
P04 = subs(P(:,:,4), d4, dc4);
P05 = subs(P(:,:,5), d5, dc5);
P06 = subs(P(:,:,6), d6, dc6);

% Defino las matrices de inercia para cada eslabón
I1 = [Ixx1 0 0; 0 Iyy1 0; 0 0 Izz1];
I2 = [Ixx2 0 0; 0 Iyy2 0; 0 0 Izz2];
I3 = [Ixx3 0 0; 0 Iyy3 0; 0 0 Izz3];
I4 = [Ixx4 0 0; 0 Iyy4 0; 0 0 Izz4];
I5 = [Ixx5 0 0; 0 Iyy5 0; 0 0 Izz5];
I6 = [Ixx6 0 0; 0 Iyy6 0; 0 0 Izz6];

% Calculo la energía cinética de cada eslabón
V1_Total = V1 + cross(W1, P01);
K1 = (1/2 * m1 * (V1_Total)') * (V1_Total) + (1/2 * W1') * (I1 * W1);
disp('Energía Cinética en el Eslabón 1');
```

```
Energía Cinética en el Eslabón 1
```

```
K1 = simplify(K1);
pretty(K1);
```

```
              2
Izz1 |th1p(t)|
---------------
       2
```

```
V2_Total = V2 + cross(W2, P02);
K2 = (1/2 * m2 * (V2_Total)') * (V2_Total) + (1/2 * W2') * (I2 * W2);
disp('Energía Cinética en el Eslabón 2');
```

```
Energía Cinética en el Eslabón 2
```

```
K2 = simplify(K2);
pretty(K2);
```

```
                                                                   /          2     2            2      2 \
                                                                   | |th1p(t)|  |d2|    |th2p(t)|  |dc2|  |
              2                  2   m2 (d2 th1p(t) - dc2 th2p(t)) | ---------------- - ----------------- |
Ixx2 |th2p(t)|    Izz2 |th1p(t)|                                   \    d2 th1p(t)         dc2 th2p(t)    /
--------------- + --------------- + ---------------------------------------------------------------------------
       2                 2                                              2
```

```
V3_Total = V3 + cross(W3, P03);
K3 = (1/2 * m3 * (V3_Total)') * (V3_Total) + (1/2 * W3') * (I3 * W3);
disp('Energía Cinética en el Eslabón 3');
```

```
Energía Cinética en el Eslabón 3
```

```
K3 = simplify(K3);
```

```
pretty(K3);
```

```
                                                 /          2           2             2  \
                                                 | |th3p(t)|     #2 |d2|      #1 |lc3|   |
                              2             2   m3 | ---------- - ---------- + ----------- | (th3p(t) - d2
Ixx3 #1   Izz3 #2   m3 #2 |l3|    m3 #1 |l3|       \   th3p(t)    d2 th1p(t)   lc3 th2p(t) /
------- + ------- + ----------- + ----------- + ----------------------------------------------------------
   2         2           2             2                                                 2

where

                  2
   #1 == |th2p(t)|

                  2
   #2 == |th1p(t)|
```

```
V4_Total = V4 + cross(W4, P04);
K4 = (1/2 * m4 * (V4_Total)') * (V4_Total) + (1/2 * W4') * (I4 * W4);
disp('Energía Cinética en el Eslabón 4');
```

```
Energía Cinética en el Eslabón 4
```

```
K4 = simplify(K4);
pretty(K4);
```

```
                                                                        /          2                \      /
                                                                        |  #4 |dc4|         #3 #1     |      |
                              m4 (th1p(t) (d4 + l3) - dc4 th4p(t)) | ----------- - ------------ |   m4 |
Ixx4 #2   Iyy4 #4   Izz4 #3                                        \ dc4 th4p(t)   d4 l3 th1p(t) /      \
------- + ------- + ------- - ------------------------------------------------------------------- + -----
   2         2         2                                         2

where

                2          2
   #1 == l3 |d4|  + d4 |l3|

                  2
   #2 == |th2p(t)|

                  2
   #3 == |th1p(t)|

                  2
   #4 == |th4p(t)|
```

```
V5_Total = V5 + cross(W5, P05);
K5 = (1/2 * m5 * (V5_Total)') * (V5_Total) + (1/2 * W5') * (I5 * W5);
disp('Energía Cinética en el Eslabón 5');
```

```
Energía Cinética en el Eslabón 5
```

```
K5 = simplify(K5);
pretty(K5);
```

```
    _______    _______  __    _______  __    _______  ___
m5 (th3p(t) - th1p(t) d2 + th2p(t) d5 + th2p(t) dc5) (th3p(t) - d2 th1p(t) + d5 th2p(t) + dc5 th2p(t))
```

```
---------------------------------------------------------------------------------------------------- + I
                                                  2
```

```
V6_Total = V6 + cross(W6, P06);
K6 = (1/2 * m6 * (V6_Total)') * (V6_Total) + (1/2 * W6') * (I6 * W6);
disp('Energía Cinética en el Eslabón 6');
```

```
Energía Cinética en el Eslabón 6
```

```
K6 = simplify(K6);
pretty(K6);
```

```
                                                                           _______   ___   ________   ________   _
m6 (th3p(t) - th1p(t) (d2 + d6) + d5 th2p(t) - d6 th5p(t) + dc6 #1) (th3p(t) + dc6 (th2p(t) + th6p(t)) + t
-----------------------------------------------------------------------------------------------------------
                                                                         2

where

   #1 == th2p(t) + th6p(t)
```

```
% Sumo todas las energías cinéticas
K_Total = simplify(K1 + K2 + K3 + K4 + K5 + K6);
disp('Energía Cinética Total');
```

```
Energía Cinética Total
```

```
pretty(K_Total);
```

```
                                                                                   m4 (#2 - dc4 th4p(t))
Ixx2 #1   Ixx3 #1   Ixx4 #1   Iyy4 #9   Izz1 #13   Izz2 #13   Izz3 #13   Izz4 #13
------- + ------- + ------- + ------- + -------- + -------- + -------- + -------- - ---------------------
   2         2         2         2          2          2          2          2

where

                  2
   #1 == |th2p(t)|

   #2 == th1p(t) (d4 + l3)

                2          2
   #3 == l3 |d4|  + d4 |l3|

         _______   _______
         th1p(t)   th5p(t)
   #4 == ------- + -------
            2         2

         _______  __   __
   #5 == th1p(t) (d4 + l3)

                  2
         |th3p(t)|
   #6 == ----------
           th3p(t)
```

```
        _______ __
#7 == th4p(t) d5

        _______ __
#8 == th2p(t) d5

                 2
#9 == |th4p(t)|

#10 == th2p(t) + th6p(t)

#11 == th1p(t) + th5p(t)

                  2
        #13 |d2|
#12 == ----------
       d2 th1p(t)

                  2
#13 == |th1p(t)|
```

5 Cálculo de la Energía Potencial

1. Se selecciona la componente vertical (por ejemplo, zzz o yyy, según la convención) como la altura hih_ihi del centro de masa de cada eslabón.
2. La energía potencial para cada eslabón es Ui=mi g hi
3. Se suma para obtener UTotal=∑Ui

```
%-------------------------------------------------------------------------
% Cálculo de la Energía Potencial
%-------------------------------------------------------------------------
% Uso las componentes verticales que elija para la altura
h1 = P01(3);
h2 = P02(2);
h3 = P03(3);
h4 = P04(3);
h5 = P05(2);
h6 = P06(3);

U1 = m1 * g * h1;
U2 = m2 * g * h2;
U3 = m3 * g * h3;
U4 = m4 * g * h4;
U5 = m5 * g * h5;
U6 = m6 * g * h6;

U_Total = U1 + U2 + U3 + U4 + U5 + U6;
```

6 Obtención del Lagrangiano y la Función de Energía

1. El Lagrangiano se define como: L=KTotal−UTotal.
2. La Función de Energía Total es: H=KTotal+UTotal.

```
%--------------------------------------------------------------------------
% Cálculo del Lagrangiano y Modelo de Energía
%--------------------------------------------------------------------------
Lagrangiano = simplify(K_Total - U_Total);
disp('Lagrangiano L = K - U');
```

```
Lagrangiano L = K - U
```

```
pretty(Lagrangiano);
```

```
                                                                                  m4 (#2 - dc4 th4p(t))
Ixx2 #1   Ixx3 #1   Ixx4 #1   Iyy4 #9   Izz1 #13   Izz2 #13   Izz3 #13   Izz4 #13
------- + ------- + ------- + ------- + -------- + -------- + -------- + -------- - ---------------------
   2         2         2         2          2          2          2          2

where

                  2
   #1 == |th2p(t)|

   #2 == th1p(t) (d4 + l3)

                2           2
   #3 == l3 |d4|  + d4 |l3|

         _______   _______
         th1p(t)   th5p(t)
   #4 == ------- + -------
            2         2

         _______  __   __
   #5 == th1p(t) (d4 + l3)

                  2
         |th3p(t)|
   #6 == ----------
           th3p(t)

         _______ __
   #7 == th4p(t) d5

         _______ __
   #8 == th2p(t) d5

                  2
   #9 == |th4p(t)|

   #10 == th2p(t) + th6p(t)

   #11 == th1p(t) + th5p(t)

                   2
           #13 |d2|
   #12 == ----------
          d2 th1p(t)

                   2
   #13 == |th1p(t)|
```

```
H = simplify(K_Total + U_Total);
disp('Función de Energía Total H = K + U');
```

```
Función de Energía Total H = K + U
```

```
pretty(H);
```

```
                                                                                     m4 (#2 - dc4 th4p(t))
Ixx2 #1   Ixx3 #1   Ixx4 #1   Iyy4 #9   Izz1 #13   Izz2 #13   Izz3 #13   Izz4 #13
------- + ------- + ------- + ------- + -------- + -------- + -------- + -------- - ---------------------
   2         2         2         2          2          2          2          2

where

                     2
   #1 == |th2p(t)|

   #2 == th1p(t) (d4 + l3)

                   2           2
   #3 == l3 |d4|  + d4 |l3|

         _______   _______
         th1p(t)   th5p(t)
   #4 == ------- + -------
            2         2

         _______  __   __
   #5 == th1p(t) (d4 + l3)

                     2
         |th3p(t)|
   #6 == ----------
           th3p(t)

         _______ __
   #7 == th4p(t) d5

         _______ __
   #8 == th2p(t) d5

                     2
   #9 == |th4p(t)|

   #10 == th2p(t) + th6p(t)

   #11 == th1p(t) + th5p(t)

                      2
           #13 |d2|
   #12 == ----------
          d2 th1p(t)

                      2
   #13 == |th1p(t)|
```

```
toc;  % Fin del conteo de tiempo
```

```
Elapsed time is 8.654818 seconds.
```

```
%-----------------------------------------------------------------------
% Funciones Auxiliares para Rotaciones
%-----------------------------------------------------------------------
function Rx = rotacion_x(theta)
    % Matriz de rotación alrededor de X en grados
    Rx = [1,         0,           0;
          0, cosd(theta), -sind(theta);
          0, sind(theta),  cosd(theta)];
end

function Rz = rotacion_z(theta)
    % Matriz de rotación alrededor de Z en grados
    Rz = [cosd(theta), -sind(theta), 0;
          sind(theta),  cosd(theta), 0;
                 0,            0, 1];
end
```

6. Conclusión

1.Eslabón 3 con Velocidad Angular No Cero:

- El hecho de que la junta 3 sea prismática solo indica que esa junta no aporta rotación.
- Sin embargo, el eslabón 3 está montado sobre juntas rotacionales (1 y 2), de manera que hereda la velocidad angular generada por esas juntas anteriores.
- Por eso, W3 aparece con componentes dependientes de $\theta 1'$ y $\theta 2'$.

2. Energía Cinética y Potencial:

- Se obtienen sumando las contribuciones traslacionales y rotacionales de cada eslabón, y considerando la altura (componente vertical) del centro de masa para la potencial.

3. Lagrangiano:

- Con $L=K-U$, se tiene la base para cualquier análisis dinámico adicional.

Con este procedimiento, se cumple el objetivo de obtener la energía total y el Lagrangiano, y se aclara la razón por la cual el eslabón 3 posee una velocidad angular que no es cero a pesar de que su junta sea prismática.